

# E-POOL, Carsharing and Carpooling Service

---

## Gruppo "MAI ESS KEW ELL "

Giacomo Minello, Matteo Tramontano, Davide Menetto

26 aprile 2019

## 1 INTRODUZIONE

IL PROGETTO E-POOL, assegnato per il corso di Basi di Dati 2018/2019, consiste nella progettazione ed implementazione di sistema informativo con interfaccia web al fine di gestire i dati di una piattaforma di carsharing e carpooling.

IL BACKEND del progetto si basa sullo stack XAMPP e in aggiunta il DBMS MongoDB per il log delle operazioni di inserimento. Per il frontend ci siamo avvalsi delle tecnologie più comunemente utilizzate (HTML+JS/JQuery).

NONOSTANTE il focus del progetto sia la realizzazione della base di dati abbiamo prestato attenzione anche alla UX, alla UI e all'implementazione di un backend di qualità. In questi ultimi ambiti riteniamo ci siano grandi margini di miglioramento ma considerando che il tempo impiegato per curare questi aspetti ha contribuito a migliorare l'usabilità ed abbia facilitato la verifica della correttezza dell'implementazione della base di dati riteniamo a posteriori che la scelta sia stata ottimale.

## 2 ANALISI DELLE SPECIFICHE

La traccia completa del progetto è disponibile a [questo indirizzo](#).

## 2.1 OPERAZIONI

Nel progetto vengono gestite le seguenti operazioni per tutti i tipi di utente:

- Registrazione di un utente
- Registrazione di un utente aziendale
- Autenticazione di un utente
- Autenticazione di un utente aziendale
- Visualizzare i dati di altri utenti
- Effettuare una prenotazione di un veicolo
- Visualizzare lo storico delle prenotazioni effettuate
- Visualizzare i veicoli disponibili
- Visualizzare i dati delle società di car-sharing
- Visualizzare i dati delle aziende
- Inserire una segnalazione sullo stato di un veicolo

Gli utenti premium o dipendenti di un'azienda possono eseguire anche le seguenti operazioni:

- Inserire un tragitto e condividerlo
- Visualizzare i tragitti prenotabili
- Aggiungersi ad una prenotazione esistente
- Inserire una valutazione di un utente con cui si è condiviso un viaggio

A tutti gli utenti sarà possibile accedere alle seguenti operazioni sui dati:

- Visualizzare la classifica degli utenti sulla base del voto medio ricevuto
- Visualizzare la classifica degli utenti più attivi in base al numero di segnalazioni
- Visualizzare la classifica dei modelli di veicolo più prenotati

## 2.2 VOLUMI

<i>Entità</i>	<i>Volume</i>
Azienda	10
Foto	1000
Passaggio	100
Pdf	200
Prenotazione	1000
Privata	10
Pubblica	10
Segnalazione	1000
Società	10
Sosta	100
Tappa	1000
Tragitto	100
Tragitto Prenotazione	10000
Utente	500
Utente Aziendale	100
Utente Premium	100
Utente Semplice	300
Valutazione	500
Veicolo	200



### 3.1.2 DIZIONARIO ENTITÀ E RELAZIONI

<i>Entità</i>	<i>Attributi</i>	<i>Identificatore</i>
azienda	NOME, INDIRIZZO, TELEFONO, RECAPITO	NOME
foto	IDFOTO, EMAIL-UTENTE, PATHFOTO	IDFOTO
passaggio	ID-PASSAGGIO, ID-TAPPA, EMAILP, EMAILA, INDIRIZZO-PARTENZA, INDIRIZZO-ARRIVO	ID-PASSAGGIO
pdf	IDPDF, NOME-SOCIETA, PATH	IDPDF
prenotazione	ID, INIZIO, FINE, NOTE, AUTO, UTENTE, INDIRIZZO-PARTENZA, INDIRIZZO-ARRIVO	ID
privata	NOME	NOME
pubblica	NOME	NOME
segnalazione	ID, EMAIL, SOCIETA, DATA, TITOLO, TESTO, AUTO	ID
societa	NOME, URL, TELEFONO	NOME
sosta	INDIRIZZO, LAT, LNG, RICARICA	INDIRIZZO
tappa	ID-TRAGITTO, CITTA, VIA, LAT	ID-TRAGITTO, VIA
tragitto	ID, EMAILP, EMAILA, KM, TIPO	ID
tragitto-prenotazione	ID-TRAG, ID-PREN	ID-TRAG, ID-PREN
utente	EMAIL, PW, NOME, COGNOME, DATANASCITA, LUOGO	EMAIL
utente-aziendale	EMAILA, NOMEAZIENDA	EMAILA
utente-premium	EMAILP	EMAILP
utente-semplce	EMAILS	EMAILS
valutazione	ID, EMAIL, DATA, TESTO, VOTO, UTENTE	ID
veicolo	TARGA, MODELLO, CAPIENZA, DESCRIZIONE, FERIALE, FESTIVO, SOCIETA, AREA-SOSTA, STATO	TARGA

<i>Relazione</i>	<i>Descrizione</i>	<i>Molteplicità</i>	<i>Componenti</i>
Disposizione	Associa n foto ad un utente	(1N)-(11)	UTENTE-FOTO
EffettuazioneU	Associa n segnalazioni ad un utente	(1N-11)	UTENTE-SEGNALAZIONE
EffettuazioneP	Associa n prenotazioni ad un utente	(1N-11)	UTENTE-PRENOTAZIONE
Assegnazione	Associa n prenotazioni ad un veicolo	(11-1N)	PRENOTAZIONE-VEICOLO
Attestazione	Associa n segnalazioni ad un veicolo	(11-1N)	SEGNALAZIONE-VEICOLO
Partenza	Associa n prenotazioni ad un area di sosta	(11-1N)	PRENOTAZIONE-AREA SOSTA
Arrivo	Associa n prenotazioni ad un area di sosta	(11-1N)	PRENOTAZIONE-AREA SOSTA
Valutazione	Associa un veicolo ad una tariffa	(11-11)	VEICOLO-TARIFFA
Collocazione	Associa n veicoli ad un area di sosta	(11-1N)	VEICOLO-AREA SOSTA
Possesso	Associa n veicoli ad una società	(11-1N)	VEICOLO-SOCIETA'
EffettuazioneS	Associa n segnalazioni ad una società	(11-1N)	SEGNALAZIONE-SOCIETA'
Pubblicizzazione	Associa n brochure ad una società privata	(1N-11)	SOCIETA' PRIVATA-BROCHURE
CondivisioneA	Associa n passaggi ad un utente aziendale	(1N-11)	UTENTE AZIENDALE-PASSAGGIO
Dipendenza	Associa n utenti aziendali ad una azienda	(11-1N)	UTENTE AZIENDALE-AZIENDA
Partenza	Associa n tappe ad un passaggio	(11-1N)	PASSAGGIO-TAPPA
Arrivo	Associa n tappe ad un passaggio	(11-1N)	PASSAGGIO-TAPPA
CondivisioneP	Associa n passaggi ad un utente premium	(1N-11)	UTENTE PREMIUM-PASSAGGIO
Ricezione	Associa n valutazioni ad un utente premium	(0N-11)	UTENTE PREMIUM-VALUTAZIONE
Esecuzione	Associa n valutazioni ad un utente premium	(0N-11)	UTENTE PREMIUM-VALUTAZIONE
CpAziendale	Associa n tragitti ad un utente aziendale	(1N-11)	UTENTE AZIENDALE-TRAGITTO
CpPremium	Associa n tragitti ad un utente premium	(1N-11)	UTENTE PREMIUM-TRAGITTO
Disposizione	Associa n tappe ad un tragitto	(11-1N)	TAPPA-TRAGITTO
TragittoPrenotazione	Associa uno o zero tragitti ad una prenotazione	(11-01)	PRENOTAZIONE-TRAGITTO

### 3.1.3 BUSINESS RULES

Vengono considerate le seguenti business rules:

1. La prenotazione è possibile solo su veicoli non già prenotati da altri utenti per la stessa data/orario
2. Tenere traccia dell'ordine delle tappe nel tragitto
3. Condivisione spese per una specifica prenotazione
4. Richiesta di condivisione accettata automaticamente a patto che non si ecceda la capacità del veicolo
5. Ogni tragitto può essere di tipo: URBANO, EXTRA-URBANO, AUTOSTRADALE, MISTO
6. Ogni area di sosta può essere dotata o meno di una colonnina di ricarica

## 3.2 PROGETTAZIONE LOGICA

### 3.2.1 RISTRUTTURAZIONE SCHEMA CONCETTUALE

Tutte le relazioni uno-a-molti sono state trasformate in attributi. Relazioni di altro tipo sono state tradotte come tabelle. Le generalizzazioni sono state tradotte con lo scopo di minimizzare i valori *null* a discapito del numero di tabelle.

### 3.2.2 RIDONDANZE

Assumendo i seguenti volumi:

- Numero medio società -> 10
- Numero medio veicoli ->100
- Numero medio veicoli per società -> 20

Assumendo il seguente peso delle operazioni:

- Peso operazioni interattive -> 0.8 ( $W_i$ )
- Peso operazioni batch -> 0.2 ( $W_b$ )
- Peso operazioni scrittura -> 2 ( $\alpha$ )

Assumendo le seguenti operazioni sui dati:

- Inserire nuovo veicolo per una specifica società -> 1 volta al mese, interattiva (OP 1)
- Rimuovere un veicolo per una specifica società -> 1 volta al mese, interattiva (OP 2)
- Contare i veicoli per una specifica società -> 5 volte al mese, batch (OP 3)

Senza ridondanza si ha che:

<i>op</i>	<i>frequenza</i>	<i>peso</i>	<i>letture</i>	<i>scritture</i>	<i>costo</i>
1	1	0.8	0	2	3.2
2	1	0.8	0	2	3.2
3	5	0.2	40	0	40

con un costo totale pari a

$$C(S) = 46.4 \quad (3.1)$$

Considerando la ridondanza invece si ha che:

<i>op</i>	<i>frequenza</i>	<i>peso</i>	<i>letture</i>	<i>scritture</i>	<i>costo</i>
1	1	0.8	0	2	3.2
2	1	0.8	0	2	3.2
3	5	0.2	1	0	1

con un costo totale pari a

$$C(Srid) = 7.2 \quad (3.2)$$

Procedendo a calcolare lo speedup si ha che


$$\frac{C(S)}{C(Srid)} = \frac{46.4}{7.2} = 6.44 \quad (3.3)$$

Questo valore di SpeedUp indica che è lievemente conveniente introdurre il campo numero veicoli nella tabella SOCIETA'






### 3.2.3 VINCOLI

La documentazione completa del database è disponibile in formato HTML negli allegati alla cartella "epool@localhost".





**azienda**


▼
Columns

	Key	Name	Data type	Null	Attributes	References	Description
	1	 NOME	varchar(30)				
	2	INDIRIZZO	varchar(50)				
	3	TELEFONO	int(10, 0)				
	4	RECAPITO	int(10, 0)				

▼
Unique keys

	Key name	Columns	Description
	PRIMARY	NOME	

**Figura 3.2:** Struttura e vincoli della tabella AZIENDA



**foto**





Documentation

epool@localhost


Name

foto


▼ Columns

	Key	Name	Data type	Null	Attributes	References	Description
	1	 IDFOTO	smallint(5, 0)		Identity		
	2	EMAIL_UTENTE	varchar(30)			<a href="#">utente</a>	
	3	PATHFOTO	varchar(100)				

▼ Relations

Foreign table	Primary table	Join	Title / Name / Description
foto	 utente	foto.EMAIL_UTENTE = utente.EMAIL	foto_ibfk_1

▼ Unique keys

Key name	Columns	Description
 PRIMARY	IDFOTO	

**Figura 3.3:** Struttura e vincoli della tabella FOTO

passaggio

Columns

	Key	Name	Data type	Null	Attributes	References	Description
	1	ID_PASSAGGIO	smallint(5, 0)		Identity		
	2	ID_TAPPA	smallint(5, 0)	✓		tappa	
	3	EMAILP	varchar(30)	✓		utente_premium	
	4	EMAILA	varchar(30)	✓		utente_aziendale	
	5	INDIRIZZO_PARTENZA	varchar(30)				
	6	INDIRIZZO_ARRIVO	varchar(30)				

Relations

Foreign table		Primary table	Join	Title / Name / Description
passaggio		tappa	passaggio.ID_TAPPA = tappa.ID_TRAGITTO	passaggio_ibfk_1
passaggio		utente_aziendale	passaggio.EMAILA = utente_aziendale.EMAILA	passaggio_ibfk_3
passaggio		utente_premium	passaggio.EMAILP = utente_premium.EMAILP	passaggio_ibfk_2

Unique keys

Key name	Columns	Description
PRIMARY	ID_PASSAGGIO	

Triggers

Key name	When	Description
AggiornaPostiDisponibili	After Insert	

**Figura 3.4:** Struttura e vincoli della tabella PASSAGGIO

pdf

Documentation

epool@localhost

Name

pdf

Columns

	Key	Name	Data type	Null	Attributes	References	Description
	1		IDPDF	smallint(5, 0)	Identity		
	2	NOME_SOCIETA	varchar(30)			pubblica	
	3	PATH	varchar(100)				

Relations


Foreign table		Primary table	Join	Title / Name / Description
pdf		pubblica	pdf.NOME_SOCIETA = pubblica.NOME	pdf_ibfk_1

Unique keys

Key name	Columns	Description
	PRIMARY	IDPDF

**Figura 3.5:** Struttura e vincoli della tabella PDF

**Figura 3.6:** Struttura e vincoli della tabella PRENOTAZIONE



# privata

Documentation



epool@localhost

Name

privata


▼

Columns

	Key	Name	Data type	Null	Attributes	References	Description
	1		NOME	varchar(30)			

▼

Unique keys

	Key name	Columns	Description
	PRIMARY	NOME	

**Figura 3.7:** Struttura e vincoli della tabella PRIVATA

pubblica

Documentation

epool@localhost

Name

pubblica

Columns

	Key	Name	Data type	Null	Attributes	References	Description
<div><div></div><div>1</div><div></div></div>		NOME	varchar(30)				

Relations

Foreign table		Primary table	Join	Title / Name / Description
pdf	<div></div>	pubblica	pdf.NOME_SOCIETA = pubblica.NOME	pdf_ibfk_1








Unique keys

	Key name	Columns	Description
<div></div>	PRIMARY	NOME	




**Figura 3.8:** Struttura e vincoli della tabella PUBBLICA

segnalazione


Columns

	Key	Name	Data type	Null	Attributes	References	Description
	1	ID	smallint(5, 0)		Identity		
	2	EMAIL	varchar(30)	✓		<a href="#">utente</a>	
	3	SOCIETA	varchar(30)	✓		<a href="#">societa</a>	
	4	DATA	date				
	5	TITOLO	varchar(20)				
	6	TESTO	varchar(200)				
	7	AUTO	varchar(10)			<a href="#">veicolo</a>	

Relations

Foreign table		Primary table	Join	Title / Name / Description
segnalazione		<a href="#">societa</a>	<b>segnalazione.SOCIETA</b> = <b>societa.NOME</b>	segnalazione_ibfk_3
segnalazione		<a href="#">utente</a>	<b>segnalazione.EMAIL</b> = <b>utente.EMAIL</b>	segnalazione_ibfk_2
segnalazione		<a href="#">veicolo</a>	<b>segnalazione.AUTO</b> = <b>veicolo.TARGA</b>	segnalazione_ibfk_1

Unique keys

Key name	Columns	Description
 PRIMARY	ID	

**Figura 3.9:** Struttura e vincoli della tabella SEGNALAZIONE

societa

Documentation

epool@localhost

Name

societa

▼ Columns

	Key	Name	Data type	Null	Attributes	References	Description
<div></div>	1	<div></div> NOME	varchar(30)				
<div></div>	2	URL	varchar(30)				
<div></div>	3	TELEFONO	int(10, 0)				

▼ Relations

Foreign table		Primary table	Join	Title / Name / Description
segnalazione	<div></div>	societa	segnalazione.SOCIETA = <b>societa</b> .NOME	segnalazione_ibfk_3
veicolo	<div></div>	societa	veicolo.SOCIETA = <b>societa</b> .NOME	veicolo_ibfk_1

▼ Unique keys

Key name	Columns	Description
<div></div> PRIMARY	NOME	

**Figura 3.10:** Struttura e vincoli della tabella SOCIETA

sosta

Documentation

epool@localhost

Name

sosta

▼ Columns

	Key	Name	Data type	Null	Attributes	References	Description
<div></div>	1	<div></div> INDIRIZZO	varchar(30)				
<div></div>	2	LAT	decimal(10, 6)				
<div></div>	3	LNG	decimal(10, 6)				
<div></div>	4	RICARICA	varchar(2)	<div></div>			

▼ Relations

Foreign table		Primary table	Join	Title / Name / Description
prenotazione	<div></div>	sosta	prenotazione.INDIRIZZO_PARTENZA = <b>sosta</b> .INDIRIZZO	prenotazione_ibfk_1
prenotazione	<div></div>	sosta	prenotazione.INDIRIZZO_ARRIVO = <b>sosta</b> .INDIRIZZO	prenotazione_ibfk_2

▼ Unique keys

Key name	Columns	Description
<div></div> PRIMARY	INDIRIZZO	

**Figura 3.11:** Struttura e vincoli della tabella SOSTA

tappa							
Columns							
	Key	Name	Data type	Null	Attributes	References	Description
1	PRIMARY	ID_TRAGITTO	smallint(5, 0)			tragitto	
2		CITTA	varchar(20)				
3		VIA	varchar(30)				
4		LAT	float(10, 6)				
5		LNG	float(10, 6)				
6		ORARIO_ARRIVO	datetime				
7		POSTI	smallint(5, 0)	✓			
Relations							
Foreign table		Primary table	Join	Title / Name / Description			
tappa	➤	tragitto	tappa.ID_TRAGITTO = tragitto.ID	tappa_ibfk_1			
passaggio	➤	tappa	passaggio.ID_TAPPA = tappa.ID_TRAGITTO	passaggio_ibfk_1			
Unique keys							
Key name	Columns	Description					
PRIMARY	ID_TRAGITTO, VIA						

Figura 3.12: Struttura e vincoli della tabella TAPPA

tragitto							
Columns							
	Key	Name	Data type	Null	Attributes	References	Description
1	PRIMARY	ID	smallint(5, 0)		Identity		
2		EMAILP	varchar(30)	✓		utente_premium	
3		EMAILA	varchar(30)	✓		utente_aziendale	
4		KM	smallint(5, 0)				
5		TIPO	enum	✓			
Relations							
Foreign table		Primary table	Join	Title / Name / Description			
tragitto	➤	utente_aziendale	tragitto.EMAILA = utente_aziendale.EMAILA	tragitto_ibfk_2			
tragitto	➤	utente_premium	tragitto.EMAILP = utente_premium.EMAILP	tragitto_ibfk_1			
tappa	➤	tragitto	tappa.ID_TRAGITTO = tragitto.ID	tappa_ibfk_1			
tragitto_prenotazione	➤	tragitto	tragitto_prenotazione.ID_TRAG = tragitto.ID	tragitto_prenotazione_ibfk_1			
Unique keys							
Key name	Columns	Description					
PRIMARY	ID						

Figura 3.13: Struttura e vincoli della tabella TRAGITTO

tragitto\_prenotazione

Documentation

epool@localhost

Name

tragitto\_prenotazione

▼

Columns

	Key	Name	Data type	Null	Attributes	References	Description
<div></div>	1	<div></div> ID_TRAG	smallint(5, 0)			tragitto	
<div></div>	2	<div></div> ID_PREN	smallint(5, 0)			prenotazione	

▼

Relations


Foreign table		Primary table	Join	Title / Name / Description
tragitto_prenotazione	➤	prenotazione	tragitto_prenotazione.ID_PREN = prenotazione.ID	tragitto_prenotazione_ibfk_2
tragitto_prenotazione	➤	tragitto	tragitto_prenotazione.ID_TRAG = tragitto.ID	tragitto_prenotazione_ibfk_1

▼

Unique keys









Key name	Columns	Description
<div></div> PRIMARY	ID_TRAG, ID_PREN	

**Figura 3.14:** Struttura e vincoli della tabella TRAGITTOPRENOTAZIONE








utente


Columns

	Key	Name	Data type	Null	Attributes	References	Description
	1	 EMAIL	varchar(30)				
	2	PW	varchar(30)				
	3	NOME	varchar(30)				
	4	COGNOME	varchar(30)				
	5	DATANASCITA	date				
	6	LUOGO	varchar(50)				

Relations

Foreign table		Primary table	Join	Title / Name / Description
foto		utente	foto.EMAIL_UTENTE = utente.EMAIL	foto_ibfk_1
prenotazione		utente	prenotazione.UTENTE = utente.EMAIL	prenotazione_ibfk_3
segnalazione		utente	segnalazione.EMAIL = utente.EMAIL	segnalazione_ibfk_2
utente_aziendale		utente	utente_aziendale.EMAILA = utente.EMAIL	utente_aziendale_ibfk_1
valutazione		utente	valutazione.UTENTE = utente.EMAIL	valutazione_ibfk_2

Unique keys

Key name	Columns	Description
 PRIMARY	EMAIL	

**Figura 3.15:** Struttura e vincoli della tabella UTENTE



utente\_aziendale

Documentation

epool@localhost

Name

utente\_aziendale

▼ Columns

	Key	Name	Data type	Null	Attributes	References	Description
	1		EMAILA	varchar(30)		<a href="#">utente</a>	
	2	NOMEAZIENDA	varchar(30)	✓			

▼ Relations

Foreign table		Primary table	Join	Title / Name / Description
utente_aziendale		<a href="#">utente</a>	utente_aziendale.EMAILA = utente.EMAIL	utente_aziendale_ibfk_1
<a href="#">passaggio</a>		utente_aziendale	passaggio.EMAILA = <a href="#">utente_aziendale</a> .EMAILA	passaggio_ibfk_3
<a href="#">tragitto</a>		utente_aziendale	tragitto.EMAILA = <a href="#">utente_aziendale</a> .EMAILA	tragitto_ibfk_2

▼ Unique keys

Key name	Columns	Description
	PRIMARY	EMAILA

**Figura 3.16:** Struttura e vincoli della tabella UTENTEAZIENDALE

utente\_premium

Documentation

epool@localhost

Name

utente\_premium

▼ Columns

	Key	Name	Data type	Null	Attributes	References	Description
<div><div></div><div>1</div><div></div></div>		EMAILP	varchar(30)				


▼ Relations

Foreign table		Primary table	Join	Title / Name / Description
passaggio	<div></div>	utente_premium	passaggio.EMAILP = utente_premium.EMAILP	passaggio_ibfk_2
tragitto	<div></div>	utente_premium	tragitto.EMAILP = utente_premium.EMAILP	tragitto_ibfk_1
valutazione	<div></div>	utente_premium	valutazione.EMAIL = utente_premium.EMAILP	valutazione_ibfk_1

▼ Unique keys

Key name	Columns	Description
<div></div> PRIMARY	EMAILP	

**Figura 3.17:** Struttura e vincoli della tabella UTENTEPREMIUM



# utente\_semplice

Documentation



epool@localhost

Name

utente\_semplice


▼

Columns

	Key	Name	Data type	Null	Attributes	References	Description
	1		EMAILS	varchar(30)			

▼

Unique keys

	Key name	Columns	Description
	PRIMARY	EMAILS	

**Figura 3.18:** Struttura e vincoli della tabella UTENTESEMPLICE

veicolo

Columns

	Key	Name	Data type	Null	Attributes	References	Description
	1	TARGA	varchar(10)				
	2	MODELLO	varchar(20)				
	3	CAPIENZA	smallint(5, 0)				
	4	DESCRIZIONE	varchar(200)				
	5	FERIALE	smallint(5, 0)				
	6	FESTIVO	smallint(5, 0)				
	7	SOCIETA	varchar(30)			<a href="#">societa</a>	
	8	AREA_SOSTA	varchar(30)				
	9	STATO	enum	✓	Default: NON IN USO		

Relations

Foreign table		Primary table	Join	Title / Name / Description
veicolo		<a href="#">societa</a>	veicolo.SOCIETA = societa.NOME	veicolo_ibfk_1
<a href="#">prenotazione</a>		veicolo	prenotazione.AUTO = <b>veicolo</b> .TARGA	prenotazione_ibfk_4
<a href="#">segnalazione</a>		veicolo	segnalazione.AUTO = <b>veicolo</b> .TARGA	segnalazione_ibfk_1

Unique keys

Key name	Columns	Description
PRIMARY	TARGA	

**Figura 3.19:** Struttura e vincoli della tabella VEICOLO

valutazione							
Columns							
Key	Name	Data type	Null	Attributes	References	Description	
1	ID	smallint(5, 0)		Identity			
2	EMAIL	varchar(30)			utente_premium		
3	DATA	datetime		Default: CURRENT_TIMESTAMP			
4	TESTO	varchar(500)					
5	VOTO	smallint(5, 0)					
6	UTENTE	varchar(30)			utente		
Relations							
Foreign table	Primary table	Join	Title / Name / Description				
valutazione	utente	valutazione.UTENTE = utente.EMAIL	valutazione_ibfk_2				
valutazione	utente_premium	valutazione.EMAIL = utente_premium.EMAILP	valutazione_ibfk_1				
Unique keys							
Key name	Columns	Description					
PRIMARY	ID						

**Figura 3.20:** Struttura e vincoli della tabella VALUTAZIONE

## 4 FUNZIONALITÀ

All'accesso al sito, durante il caricamento della pagina principale, viene mostrata un'ipnotizante animazione di caricamento. Una volta caricata la pagina principale del sito l'utente ha la possibilità di inserire rapidamente le proprie credenziali per autenticarsi attraverso due form:

- il primo form è dedicato agli utenti semplici e premium
- il secondo form è dedicato agli utenti dipendenti di un'azienda

La scelta di non unificare le due modalità di autenticazione è stata fatta per dare la possibilità di separare gli accessi, immaginando una pagina di accesso dedicata per i dipendenti di un'azienda. "Under the hood" all'accesso al sito viene caricata la pagina *index* che include anche i form per il login. Ciò permette di dare spazio ad ulteriori modifiche future e permettere facilmente di riarrangiare la UX del sito, ad esempio inserendo un form per il login ed uno per la registrazione, messaggi di benvenuto, istruzioni per l'utente, ecc..

Qualora l'utente non fosse registrato è possibile accedere ai form di registrazione tramite il menu. Come per il login abbiamo scelto di separare i due metodi di registrazione, sempre tenendo conto di voler mantenere una pagina dedicata per i dipendenti di un'azienda. Qualora l'utente non abbia effettuato il login e provi ad accedere a delle pagine riservate ad utenti già autenticati egli verrà reindirizzato alla pagina principale del sito dove avrà la possibilità di autenticarsi. Allo stesso modo, qualora l'utente abbia effettuato il login e provi ad accedere a delle pagine per le quali non è autorizzato egli verrà reindirizzato alla home (questo caso

si applica in particolare agli utenti semplici che provano ad accedere a funzioni riservate ad utenti premium e ai dipendenti di un'azienda).

Una volta effettuato il login con successo l'utente viene reindirizzato alla home dedicata in base al tipo di utente. Gli utenti semplici e premium condividono la stessa home (avendo però permessi diversi) e i dipendenti di un'azienda hanno una home dedicata. Entrambe le home includono un header, un footer ed un body vuoto (dove si può immaginare vengano mostrate comunicazioni, informazioni, ecc..).

Tramite il menu è possibile visualizzare tutte le pagine del sito nella quali i nomi indicano le features disponibili. Le pagine sono raggruppate in:

- home, un link utile per tornare alla pagina principale del sito (sebbene la funzione sia disponibile cliccando il logo nello stesso menu, per ottimizzare l'UX sia in modalità desktop che mobile il logo viene nascosto quando si scorre una pagina).
- utente standard, contiene le funzioni disponibili per tutti gli utenti autenticati.
- premium features, contiene le funzioni disponibili per gli utenti premium e i dipendenti di un'azienda.
- registrazione, permette di accedere ai form di registrazione.
- statistiche, una pagina pubblica che permette di visualizzare alcuni dati relativi alla piattaforma.

Senza entrare nel dettaglio di ogni pagina alcune funzionalità non strettamente relative al database che sono state incluse sono:

- nella pagina del *profilo* utente è possibile caricare delle foto e visualizzarle su uno slideshow (una piccola nota, dopo il caricamento di una foto l'immagine potrebbe non essere immediatamente visibile, per ovviare al problema è possibile disabilitare la cache del sito in modo da forzarne l'aggiornamento).
- nella pagina *veicolidisponibili* è possibile visualizzare i veicoli e la loro posizione su una mappa tramite le API Google (di recente sono cambiate le modalità per ottenere una Key valida per l'utilizzo delle API, imponendo l'obbligo di abilitare la fatturazione di Google Cloud Platform: eventuali problemi di caricamento sono dovuti all'utilizzo di una chiave non valida che abbiamo utilizzato per limitarci a mostrare la possibilità di implementare questa funzione).
- nella pagina *visualizzasocietà* è possibile visualizzare i dati delle società di car sharing ed anche delle brochure pdf se presenti (la possibilità di inserire tali brochure non è stata gestita tramite il sito immaginando che tale funzionalità venga riservata ad un amministratore o alla società stessa; la gestione di questo caso d'uso è quindi presente come stored procedure).

## 5 CODICE MySQL

Per agevolare la leggibilità del codice SQL ne riportiamo solo alcuni snippets. Il codice completo è disponibile negli allegati.

### 5.1 VISTE

#### 5.1.1 MEDIA VOTO UTENTE

```
1 CREATE VIEW MEDIA_VOTO_UTENTE (UTENTE, MEDIA_VOTO) AS
2 SELECT UTENTE, AVG (VOTO)
3 FROM VALUTAZIONE
4 GROUP BY UTENTE
5 ORDER BY AVG(VOTO) DESC;
```

#### 5.1.2 PRENOTAZIONE CAPIENZA

```
1 CREATE VIEW PRENOTAZIONE_CAPIENZA (AUTO, POSTI_DISPONIBILI, IDTRAGITTO ,GUIDATORE,
2 TAPPA_CITTA, TAPPA_VIA) AS
3 SELECT TARGA, CAPIENZA-1, TRAGITTO.ID, UTENTE, TAPPA.CITTA, TAPPA.VIA
4 FROM PRENOTAZIONE, VEICOLO, TRAGITTO_PRENOTAZIONE, TRAGITTO, TAPPA
5 WHERE (AUTO=TARGA AND PRENOTAZIONE.ID = ID_PREN AND TRAGITTO.ID = ID_TRAG AND
6 ID_TRAGITTO= TRAGITTO.ID);
```

#### 5.1.3 VEICOLI DISPONIBILI

```
1 CREATE VIEW VEICOLI_DISPONIBILI(TARGA, MODELLO, CAPIENZA, DESCRIZIONE, FERIALE,
2 FESTIVO, SOCIETA, AREA_SOSTA) AS
3 SELECT TARGA, MODELLO, CAPIENZA, DESCRIZIONE, FERIALE, FESTIVO, SOCIETA, AREA_SOSTA
4 FROM VEICOLO
5 WHERE STATO = 'NON IN USO'
```

## 5.2 TRIGGERS

### 5.2.1 AGGIORNA POSTI DISPONIBILI

```
1 DELIMITER |
2 CREATE TRIGGER AggiornaPostiDisponibili
3 AFTER INSERT ON PASSAGGIO
4 FOR EACH ROW
5 BEGIN
6 UPDATE TAPPA
7 SET TAPPA.POSTI=TAPPA.POSTI-1
8 WHERE TAPPA.VIA=new.INDIRIZZO_ARRIVO and tappa.ID_TRAGITTO=new.ID_TAPPA;
9 END
10 |
11 DELIMITER ;
```

### 5.2.2 BLOCCA VEICOLO

```
1 DELIMITER |
2 CREATE TRIGGER BloccaVeicolo
3 AFTER INSERT ON PRENOTAZIONE
4 FOR EACH ROW
5 BEGIN
6 UPDATE VEICOLO SET STATO = 'IN USO' WHERE TARGA=NEW.AUTO;
7 END;
8 |
9 DELIMITER ;
```

### 5.2.3 AGGIORNA LIVELLO UTENTE

```
1 DELIMITER |
2 CREATE TRIGGER AggiornaLivelloUtente
3 AFTER INSERT ON PRENOTAZIONE
4 FOR EACH ROW
5 BEGIN
6 IF ((SELECT COUNT(*) FROM prenotazione WHERE UTENTE = NEW.UTENTE) = 3)
7 THEN
8 DELETE FROM utente_semplice WHERE EMAILS IN
9 (SELECT UTENTE FROM PRENOTAZIONE WHERE PRENOTAZIONE.UTENTE = NEW.UTENTE);
10 INSERT INTO utente_premium (EMAILP)
11 (SELECT distinct (UTENTE) FROM PRENOTAZIONE WHERE PRENOTAZIONE.UTENTE = NEW.UTENTE);
12 END IF;
13 END;
```

## 5.3 STORED PRECEDURES

### 5.3.1 LOGIN

```
1 DELIMITER |
2 create PROCEDURE Login (IN Email varchar(30), IN pasw varchar(30), OUT result BOOLEAN)
3 BEGIN
4 IF EXISTS(SELECT * FROM UTENTE WHERE UTENTE.EMAIL = Email AND UTENTE.PW=pasw) THEN
5 SET result = (TRUE);
6 ELSE
7 SET result = (FALSE);
8 END IF;
9 END;
10 |
11 DELIMITER ;
```

### 5.3.2 LOGIN AZIENDALE

```
1 DELIMITER |
2 create PROCEDURE LoginAziendale (IN Email varchar(30), IN pasw varchar(30),
3 OUT result BOOLEAN)
4 BEGIN
5 IF EXISTS(SELECT * FROM UTENTE WHERE UTENTE.EMAIL = Email AND UTENTE.PW=pasw) THEN
6 IF EXISTS(SELECT * FROM UTENTE_AZIENDALE WHERE UTENTE_AZIENDALE.EMAILa = Email) THEN
7 SET result = (TRUE);
8 ELSE
9 SET result = (FALSE);
10 END IF;
11 ELSE
12 SET result = (FALSE);
13 END IF;
14 END;
15 |
16 DELIMITER ;
```

### 5.3.3 LOGIN TYPE

```
1 DELIMITER |
2 create PROCEDURE LoginType (IN Email varchar(30), OUT result BOOLEAN)
3 BEGIN
4 IF EXISTS(SELECT * FROM UTENTE_PREMIUM WHERE UTENTE_PREMIUM.EMAILP = Email) THEN
5 SET result = 2;
6 ELSE
7 SET result = 1;
8 END IF;
9 END;
10 |
11 DELIMITER ;
```



#### 5.3.4 REGISTRAZIONE UTENTE

```
1 DELIMITER |
2 CREATE PROCEDURE RegistrazioneUtente (IN EmailN varchar(30), IN pasw varchar(30),
3 IN nome varchar(30), IN cognome varchar (30),
4 IN datanascita date, IN luogo varchar (50), OUT result BOOLEAN)
5 BEGIN
6 start transaction;
7 SET result = (FALSE);
8 IF NOT EXISTS ( SELECT EMAIL FROM UTENTEWHERE EMAIL = EmailN)
9 THEN
10 INSERT INTO UTENTE (EMAIL, PW, NOME, COGNOME, DATANASCITA, LUOGO)
11 VALUES (EmailN, pasw, nome, cognome, datanascita, luogo);
12 SET result = (TRUE);
13 commit work;
14 ELSE
15 CALL printf ('[ERRORE] UTENTE GIA REGISTRATO');
16 rollback;
17 END IF;
18 END;
19 |
20 DELIMITER ;
```

### 5.3.5 REGISTRAZIONE DIPENDENTE

```
1 DELIMITER |
2 CREATE PROCEDURE RegistrazioneAziendale (IN Emailt varchar(30), IN pasw varchar(30),
3 IN nome varchar(30), IN cognome varchar (30),
4 IN datanascita date, IN luogo varchar (50), IN nomeazienda varchar (30),
5 OUT result BOOLEAN)
6 BEGIN
7 start transaction;
8 SET result = (FALSE);
9 IF NOT EXISTS ( SELECT EMAIL FROM UTENTE WHERE EMAIL = Emailt)
10 THEN IF EXISTS (SELECT * FROM AZIENDA WHERE NOMEAZIENDA = nomeazienda)
11 THEN
12 INSERT INTO UTENTE (EMAIL, PW, NOME, COGNOME, DATANASCITA, LUOGO)
13 VALUES (Emailt, pasw, nome, cognome, datanascita, luogo);
14 INSERT INTO UTENTE_AZIENDALE (EMAILA,NOMEAZIENDA) VALUES (Emailt, nomeazienda);
15 SET result = (TRUE);
16 commit work;
17 ELSE
18 CALL printf ('[ERRORE] AZIENDA NON PRESENTE NEL SISTEMA');
19 rollback;
20 END IF;
21 ELSE
22 CALL printf ('[ERRORE] UTENTE GIA REGISTRATO');
23 rollback;
24 END IF;
25 END;
26 |
27 DELIMITER ;
```

### 5.3.6 INSERISCI PRENOTAZIONE PREMIUM/AZIENDALE

```
1 DELIMITER |
2 CREATE PROCEDURE InserisciPrenotazioneAziendale (IN Tragitto smallint,
3 IN Notet VARCHAR(300), IN Automobile VARCHAR(10), IN Emailt VARCHAR(30),
4 IN IndirizzoPartenzat VARCHAR(30), IN IndirizzoArrivot VARCHAR(30), OUT result BOOLEAN)
5 BEGIN
6 start transaction;
7 SET result = (FALSE);
8 INSERT INTO `prenotazione` ( `NOTE`, `AUTO`, `UTENTE`, `INDIRIZZO_PARTENZA`,
9 `INDIRIZZO_ARRIVO`)
10 VALUES ( Notet, Automobile, Emailt, IndirizzoPartenzat, IndirizzoArrivot);
11 SET result = (SELECT LAST_INSERT_ID());
12 INSERT INTO `tragitto_prenotazione` ( `ID_TRAG`, `ID_PREN`) VALUES ( Tragitto, result);
13 SET result = (TRUE);
14 commit work;
15 END;
16 |
17 DELIMITER ;
```

### 5.3.7 TERMINA PRENOTAZIONE

```
1 DELIMITER |
2 CREATE PROCEDURE `TerminaPrenotazione`(IN `Email` VARCHAR(30), IN `Auto` VARCHAR(10))
3 BEGIN
4 DECLARE TEMPO DATETIME DEFAULT NOW();
5 start transaction;
6 IF EXISTS (SELECT * FROM PRENOTAZIONE WHERE Auto = AUTO AND UTENTE = Email)
7 THEN
8 UPDATE VEICOLO SET STATO = 'NON IN USO' WHERE TARGA = Auto;
9 UPDATE VEICOLO SET AREA_SOSTA =
10 (SELECT INDIRIZZO_ARRIVO FROM PRENOTAZIONE
11 WHERE Prenotazione.AUTO = Auto AND FINE is null) WHERE TARGA = Auto;
12 UPDATE PRENOTAZIONE SET FINE = TEMPO WHERE Prenotazione.AUTO = Auto;
13 commit work;
14 ELSE
15 CALL printf ('[ERRORE] NON ESISTE UN VEICOLO DA TE PRENOTATO CON QUESTA TARGA');
16 rollback;
17 END IF;
18 END
19 |
20 DELIMITER ;
```

### 5.3.8 INSERISCI TRAGITTO

```
1 DELIMITER |
2 CREATE PROCEDURE inserisciTragitto (IN EmailN varchar (30), IN km smallint,
3 IN tipe VARCHAR (30), OUT res smallint)
4 BEGIN
5 IF EXISTS ( SELECT EMAILP FROM UTENTE_PREMIUM WHERE EMAILP = EmailN )
6 THEN INSERT INTO TRAGITTO (EMAILP, KM, TIPO) values (EmailN, km, tipe);
7 SET res = (SELECT LAST_INSERT_ID());
8 ELSE
9 CALL printf('QUALCOSA PREMIUM è ANDATO STORTO');
10 END IF;
11 IF EXISTS (SELECT EMAILA FROM UTENTE_AZIENDALE WHERE EMAILA = EmailN )
12 THEN INSERT INTO TRAGITTO (EMAILA, KM, TIPO) values (EmailN, km, tipe);
13 CALL printf ('INSERIMENTO AZIENDALE AVVENUTO');
14 SET res = (SELECT LAST_INSERT_ID());
15 ELSE
16 CALL printf('QUALCOSA AZIENDALE è ANDATO STORTO');
17 END IF;
18 END;
19 |
20 DELIMITER ;
```

### 5.3.9 INSERISCI TAPPA

```
1 DELIMITER |
2 CREATE PROCEDURE InserisciTappa (IN idN smallint, IN cittaN varchar (20),
3 IN viaN varchar (30), IN orarioN datetime)
4 BEGIN
5 DECLARE x SMALLINT;
6 SET x = (SELECT CAPIENZA FROM veicolo,tragitto_prenotazione, prenotazione
7 WHERE tragitto_prenotazione.ID_TRAG=idN AND
8 tragitto_prenotazione.ID_PREN=prenotazione.ID AND
9 veicolo.TARGA=prenotazione.AUTO
10 limit 1);
11 IF EXISTS (SELECT ID FROM TRAGITTO WHERE ID = idN)
12 THEN INSERT INTO TAPPA (ID_TRAGITTO, CITTA, VIA, ORARIO_ARRIVO, POSTI)
13 VALUES (idN, cittaN, viaN, orarioN, X-1 );
14 ELSE
15 CALL printf ('[ERRORE], NON ESISTE NESSUN TRAGITTO CON QUESTO ID');
16 END IF;
17 END;
18 |
19 DELIMITER ;
```

## 6 CODICE NoSQL

```
1 <?php
2 function mongoLog($document)
3 {
4     if (extension_loaded("mongodb")) {
5         try {
6             $mng = new MongoDB\Driver\Manager("mongodb://localhost:27017");
7             $bulk = new MongoDB\Driver\BulkWrite();
8             $document = ['_id' => new MongoDB\BSON\ObjectId(), $document];
9             $bulk->insert($document);
10            $mng->executeBulkWrite('testPHP.LOG', $bulk);
11        } catch (MongoDB\Driver\Exception\Exception $e) {
12            //Errore
13        }
14    }
15 }
16 ?>
```

Il log delle operazioni di inserimento di dati nel database è stato gestito tramite delle funzioni in PHP. La difficoltà principale nell'implementare questa funzionalità è stata la configurazione del software, in particolare l'installazione manuale del driver PHP per MongoDB senza utilizzare PEAR o PECL. Esistono infatti due versioni di tale driver: una versione legacy ben documentata compatibile con PHP 5.x ed una versione più recente compatibile con PHP 7.x ma meno documentata. Ad ogni modo, dopo aver configurato manualmente questa estensione l'implementazione è risultata relativamente semplice; le informazioni che vengono salvate nel log sono la query che è stata eseguita, ottenuta dall'oggetto PDO e il risultato dell'esecuzione (se disponibile). Questi dati vengono inseriti in un array che viene passato ad una funzione che si occupa della connessione, di inserire l'array in un documento valido e di eseguire l'inserimento del log.

Giacomo Minello



Matteo Tramontano



Davide Menetto

